日本ソフトウェア科学会第 28 回大会 (2011 年度) 講演論文集

Combinators for Streams

Koji Nakazawa

This paper proposes a new combinatory calculus CL^{μ}_{ext}, which is an extension of the untyped variant of the combinatory logic CL with variables and combinators for streams, and which is equivalent to the untyped lambda-mu calculus. This paper also proposes a class of models of the untyped lambda-mu calculus, called extensional stream models, and gives an algebraic characterization for them, which is induced by the structure of CL^{μ}_{ext}.

1 Introduction

The $\lambda\mu$-calculus is originally proposed by Parigot in [3] as a term assignment system for the classical natural deduction, and some variants of $\lambda\mu$-calculus have been widely studied as typed calculi with control operators. Parigot has noted that the μ-abstraction of the $\lambda\mu$-calculus can be seen as a potentially-infinite sequence of the λ-abstraction, and Saurin has shown that the untyped $\lambda\mu$-calculus can be seen as a stream calculus which enjoys some fundamental properties such as the separation theorem and the standardization theorem [4] [5] [6].

For the λ-calculus, it is well-known that the untyped combinatory logic CL with the combinators K and S corresponds to the untyped λ-calculus, and CL suggests an algebraic characterization of the λ-models.

This paper extends this discussion to the untyped $\lambda\mu$-calculus. The results of this paper are the following: (1) A new combinatory calculus CL^{μ}_{ext} is proposed. It is an extension of the extensional variant of CL, and exactly corresponds to the untyped $\lambda\mu$-calculus. (2) A class of models, called extensional stream models, for the untyped $\lambda\mu$-calculus is introduced. It is based on the idea that the μ-abstractions represent functions on streams. (3) An algebraic characterization for the extensional stream models is given. It is induced by the structure of CL^{μ}_{ext}, and gives another definition of the extensional stream models independent of the syntax of the $\lambda\mu$-calculus.

2 Untyped $\lambda\mu$-calculus

First, we remind the untyped $\lambda\mu$-calculus. We are following the notation of [4], because it is suitable to see the $\lambda\mu$-calculus as a calculus for streams.

Definition ($\lambda\mu$-calculus). Suppose that there are two disjoint sets of variables: one is the set Vars_T of *term variables*, denoted by $x, y, \cdots$, and the other is the set Vars_S of *stream variables*, denoted by $\alpha, \beta, \cdots$. Terms and axioms of the $\lambda\mu$-calculus are given in the Fig. 1. We use the following abbreviations: (i) $\lambda x_1 x_2 \cdots x_n.M$ denotes $\lambda x_1.(\lambda x_2.(\cdots(\lambda x_n.M)\cdots))$ (and similarly for μ), (ii) $(M)A_1 \cdots A_n$ denotes $(\cdots((M)A_1)\cdots)A_n$, in which each A_i denotes either a term or a stream variable, and the parentheses at the head position are also often omitted. Variable occurrences of x and α are bound in $\lambda x.M$ and $\mu\alpha.M$, respectively. Variable occurrences which are not bound are called free, and $FV(M)$ denotes the set of variables freely occurring in M.

In the axioms, $M[x := N]$ and $M[\alpha := \beta]$ are usual capture-avoiding substitutions, and $M[(P)\alpha := (P)N\alpha]$ recursively replaces any subterm of the form $(P)\alpha$ in M by $(P)N\alpha$. The relation $M =_{\lambda\mu} N$ is the compatible equivalence rela-

---

ストリームのための組合せ子

中澤 巧爾, 京都大学大学院情報学研究科, Graduate School of Informatics, Kyoto University.

Terms:

$$M, N ::= x \mid \lambda x.M \mid (M)N \mid \mu\alpha.M \mid (M)\alpha$$

Axioms:

$$\begin{array}{ll} (\lambda x.M)N =_{\beta_T} M[x := N] & \\ (\mu\alpha.M)\beta =_{\beta_S} M[\alpha := \beta] & \\ \lambda x.(M)x =_{\eta_T} M & (x \in FV(M)) \\ \mu\alpha.(M)\alpha =_{\eta_S} M & (\alpha \in FV(M)) \\ (\mu\alpha.M)N =_{\mu} \mu\alpha.M[(P)\alpha := (P)N\alpha] & \end{array}$$

Fig. 1 Untyped $\lambda\mu$-calculus

tion defined from the axioms.

In Parigot's original $\lambda\mu$-calculus [3], the terms of the form $(P)\alpha$, which are originally denoted by $[\alpha]P$, are distinguished as *named terms* from the ordinary terms, and bodies of μ-abstractions are restricted to the named terms. On the other hand, we consider $(P)\alpha$ as an ordinary term and any term can be the body of μ-abstraction. For example, neither $(M)\alpha N$ nor $\mu\alpha.x$ is allowed as a term in the original $\lambda\mu$-calculus, while they are well-formed terms in our calculus. This calculus is called $\Lambda\mu$-calculus by Saurin in [4], where another axiom

$$\mu\alpha.M \to_{fst} \lambda x.\mu\alpha.M[(P)\alpha := (P)x\alpha]$$

is chosen instead of (μ) for the reduction system. For equational systems with (η), the axioms (μ) and (fst) are equivalent since

$$\begin{aligned} (\mu\alpha.M)N &=_{fst} (\lambda x.\mu\alpha.M[(P)\alpha := (P)x\alpha])N \\ &=_{\beta_T} \mu\alpha.M[(P)\alpha := (P)N\alpha]), \end{aligned}$$

and

$$\begin{aligned} \mu\alpha.M &=_{\eta_T} \lambda x.(\mu\alpha.M)x \\ &=_{\mu} \lambda x.\mu\alpha.M[(P)\alpha := (P)x\alpha]. \end{aligned}$$

The untyped $\lambda\mu$-calculus can be seen as a calculus for streams, in which the μ-abstractions represent functions on stream data, and a term $(M)N_0 \cdots N_n\alpha$ means a function application of M to the stream data $N_0 \cdots N_n\alpha$, the initial segment of which is $N_0 \cdots N_n$ and the rest is α. For example, the term $\mathsf{hd} = \lambda x.\mu\alpha.x$ is the function to get the head element of streams since

$$\begin{aligned} (\mathsf{hd})N_0 \cdots N_n\beta &=_{\beta_T} (\mu\alpha.N_0)N_1 \cdots N_n\beta \\ &=_{\mu} (\mu\alpha.N_0)N_2 \cdots N_n\beta \\ \cdots &=_{\mu} (\mu\alpha.N_0)\beta \\ &=_{\beta_S} N_0. \end{aligned}$$

As a little complicated example, we have a term nth representing the function which takes a stream and a church numeral c_n and returns the n-th element of the stream. The term nth is defined as

$$Y(\lambda f x.\mu\alpha.\lambda y.\mathsf{if}\ (\mathsf{zero?}\ y)\ \mathsf{then}\ x\ \mathsf{else}\ f\alpha(y-1)),$$

where Y is a fixed point operator in the λ-calculus, and we have $\mathsf{nth}\, N_0N_1N_2 \cdots N_n\beta\, c_i =_{\lambda\mu} N_i$ for any $0 \le i \le n$. However, the $\lambda\mu$-calculus has no term representing a stream, and that means $\lambda\mu$-terms do not represent any function which returns streams.

3 Combinatory Calculus CL^{μ}_{ext}

We introduce the new combinatory calculus CL^{μ}_{ext}, and show that CL^{μ}_{ext} is equivalent to the $\lambda\mu$-calculus.

This result is an extension of the equivalence between the λ-calculus and the untyped variant of the ordinary combinatory logic CL with the combinators K and S. In CL^{μ}_{ext}, the combinators K and S are denoted by K_0 and S_0, respectively.

Definition (CL^{μ}_{ext}). Similarly to the $\lambda\mu$-calculus, CL^{μ}_{ext} has two sorts of variables: term variables Vars_T and stream variables Vars_S. Constants, terms, axioms, and extensionality rules (ζ) are given in Fig. 2. We define $FV(T)$ as the set of variables occurring in T. We suppose that $\cdot$ and $\star$ are left associative. For example, $T_1 \cdot T_2 \star \alpha \cdot T_3$ denotes $((T_1 \cdot T_2) \star \alpha) \cdot T_3$. For simplicity, we consider the following auxiliary notion of *stream terms*

$$\mathcal{S} ::= \alpha \mid T :: \mathcal{S},$$

and then the operation $\star$ is extended to the meta-operation for terms and stream terms by $T \star (T' :: \mathcal{S}) = (T \cdot T') \star \mathcal{S}$. A stream term $\mathcal{S}$ is not a term of CL^{μ}_{ext}, but $T \star \mathcal{S}$ is always a term. The meta-variables $\mathcal{S}_1, \mathcal{S}_2 \cdots$ in the axioms in Fig. 2 denote stream terms. The substitutions $T[x := T']$ and $T[\alpha := \mathcal{S}]$ are defined straightforwardly.

The relation $T =_{\mathsf{CL}^{\mu}_{ext}} U$ is the compatible equivalence relation defined from the axioms and the ex-

Constants:

$$C ::= \mathsf{K}_0 \mid \mathsf{K}_1 \mid \mathsf{S}_0 \mid \mathsf{S}_1 \mid \mathsf{C}_{1,0} \mid \mathsf{C}_{1,1} \mid \mathsf{W}_1$$

Terms:

$$T, U ::= C \mid x \mid T \cdot U \mid T \star \alpha$$

Axioms:

$$\mathsf{K}_0 \cdot T_1 \cdot T_2 = T_1 \qquad \mathsf{K}_1 \cdot T_1 \star \mathcal{S}_2 = T_1$$
$$\mathsf{S}_0 \cdot T_1 \cdot T_2 \cdot T_3 = T_1 \cdot T_3 \cdot (T_2 \cdot T_3) \qquad \mathsf{S}_1 \cdot T_1 \cdot T_2 \star \mathcal{S}_3 = T_1 \star \mathcal{S}_3 \cdot (T_2 \star \mathcal{S}_3)$$
$$\mathsf{C}_{1,0} \cdot T_1 \star \mathcal{S}_2 \cdot T_3 = T_1 \cdot T_3 \star \mathcal{S}_2 \qquad \mathsf{C}_{1,1} \cdot T_1 \star \mathcal{S}_2 \star \mathcal{S}_3 = T_1 \star \mathcal{S}_3 \star \mathcal{S}_2$$
$$\mathsf{W}_1 \cdot T_1 \star \mathcal{S}_2 = T_1 \star \mathcal{S}_1 \star \mathcal{S}_2$$

Extensionality rules:

$$\frac{T \cdot x = U \cdot x \quad x \notin FV(T) \cup FV(U)}{T = U}\ (\zeta_T) \qquad \frac{T \star \alpha = U \star \alpha \quad \alpha \notin FV(T) \cup FV(U)}{T = U}\ (\zeta_S)$$

Fig. 2 CL^{μ}_{ext}

tensionality rules.

Intuitively, stream variables and stream terms denote streams. The new operation $\star$ represents the function application for streams, which corresponds to the application $(M)\alpha$ in the $\lambda\mu$-calculus.

The correspondence between CL^{μ}_{ext} and the $\lambda\mu$-calculus can be formalized as the following translations, and the two calculi are equivalent through the translations as Theorem 1.

Definition (Translations between $\lambda\mu$ and CL^{μ}_{ext}).
1. For a CL^{μ}_{ext}-term T and a term variable x, we define the CL^{μ}_{ext}-term $\lambda^* x.T$ as follows:

$$\lambda^* x.x = \mathsf{S}_0 \cdot \mathsf{S}_0 \cdot \mathsf{K}_0$$
$$\lambda^* x.T = \mathsf{K}_0 \cdot T \qquad (x \notin FV(T))$$
$$\lambda^* x.T \cdot U = \mathsf{S}_0 \cdot (\lambda^* x.T) \cdot (\lambda^* x.U)$$
$$\lambda^* x.T \star \alpha = \mathsf{C}_{1,0} \cdot (\lambda^* x.T) \star \alpha.$$

For a CL^{μ}_{ext}-term T and a stream variable α, we define the CL^{μ}_{ext}-term $\mu^* \alpha.T$ as follows:

$$\mu^* \alpha.T = \mathsf{K}_1 \cdot T \qquad (\alpha \notin FV(T))$$
$$\mu^* \alpha.T \cdot U = \mathsf{S}_1 \cdot (\mu^* \alpha.T) \cdot (\mu^* \alpha.U)$$
$$\mu^* \alpha.T \star \alpha = \mathsf{W}_1 \cdot (\mu^* \alpha.T)$$
$$\mu^* \alpha.T \star \beta = \mathsf{C}_{1,1} \cdot (\mu^* \alpha.T) \star \beta \qquad (\alpha \neq \beta).$$

Then the mapping M^* from $\lambda\mu$-terms to CL^{μ}_{ext}-terms is defined by

$$x^* = x$$
$$(\lambda x.M)^* = \lambda^* x.M^*$$
$$((M)N)^* = M^* \cdot N^*$$
$$(\mu\alpha.M)^* = \mu^* \alpha.M^*$$
$$((M)\alpha)^* = M^* \star \alpha.$$

2. The mapping T_* from CL^{μ}_{ext}-terms to $\lambda\mu$-terms is defined by

$$(\mathsf{K}_0)_* = \lambda xy.x$$
$$(\mathsf{K}_1)_* = \lambda x.\mu\alpha.x$$
$$(\mathsf{S}_0)_* = \lambda xyz.xz(yz)$$
$$(\mathsf{S}_1)_* = \lambda xy.\mu\alpha.x\alpha(y\alpha)$$
$$(\mathsf{C}_{1,0})_* = \lambda x.\mu\alpha.\lambda y.xy\alpha$$
$$(\mathsf{C}_{1,1})_* = \lambda x.\mu\alpha\beta.x\beta\alpha$$
$$(\mathsf{W}_1)_* = \lambda x.\mu\alpha.x\alpha\alpha$$
$$x_* = x$$
$$(T \cdot U)_* = (T_*)U_*$$
$$(T \star \alpha)_* = (T_*)\alpha.$$

Lemma 1. (1) $(\lambda^* x.T) \cdot U =_{\mathsf{CL}^{\mu}_{ext}} T[x := U]$ and $(\mu^* \alpha.T) \star \mathcal{S} =_{\mathsf{CL}^{\mu}_{ext}} T[\alpha := \mathcal{S}]$.
(2) $T =_{\mathsf{CL}^{\mu}_{ext}} U$ implies $\lambda^* x.T =_{\mathsf{CL}^{\mu}_{ext}} \lambda^* x.U$ and $\mu^* \alpha.T =_{\mathsf{CL}^{\mu}_{ext}} \mu^* \alpha.U$.
(3) $M =_{\lambda\mu} N$ implies $M^* =_{\mathsf{CL}^{\mu}_{ext}} N^*$.
(4) $T =_{\mathsf{CL}^{\mu}_{ext}} U$ implies $T_* =_{\lambda\mu} U_*$.
(5) $(M^*)_* =_{\lambda\mu} M$.
(6) $(T_*)^* =_{\mathsf{CL}^{\mu}_{ext}} T$.

From these lemmas, we can prove that the combinatory calculus CL^{μ}_{ext} is equivalent to the $\lambda\mu$-calculus in the following sense.

Theorem 1. (1) For any $\lambda\mu$-terms M and N, $M =_{\lambda\mu} N$ iff $M^* =_{\mathsf{CL}^{\mu}_{ext}} N^*$.
(2) For any CL^{μ}_{ext}-terms T and U, $T =_{\mathsf{CL}^{\mu}_{ext}} U$ iff $T_* =_{\lambda\mu} U_*$.

4 Extensional Stream Models

In this section, we introduce a new class of models, called extensional stream models, for the untyped $\lambda\mu$-calculus.

First, we give a definition of the extensional stream models, straightforwardly following the idea that the $\lambda\mu$-calculus is a stream calculus. However, it depends on the definability of the interpretation of $\lambda\mu$-terms, so we give an algebraic characterization of the extensional stream models induced by the structure of CL^{μ}_{ext}, which is independent of the syntax of the $\lambda\mu$-calculus.

4.1 Definition

Let ω be the set of the natural numbers, and, for a set D, D^ω denotes the set of functions from ω to D. Each element of D^ω can be seen as a stream over D. In the following, we use $\overline{\lambda}$ to represent meta-level functions, and use the following basic functions for $d \in D$ and $s \in D^\omega$:

$$hd(s) = s(0),$$
$$tl(s) = \overline{\lambda} n \in \omega.s(n+1),$$
$$d :: s = \overline{\lambda} n \in \omega. \begin{cases} d & (n = 0) \\ s(i) & (n = i+1). \end{cases}$$

For a function $f : D \times D^\omega \to D'$, $\overline{\lambda} d :: s \in D^\omega.f(d, s)$ denotes the function $\overline{\lambda} s \in D^\omega.f(hd(s), tl(s))$.

Definition (Extensional stream models). A non-empty set D is called an *extensional stream model* if the following hold.

(1) There exists a subset $[D^\omega \to D]$ of the set of functions from D^ω to D.

(2) There exist two functions $\Phi : D \to [D^\omega \to D]$ and $\Psi : [D^\omega \to D] \to D$ such that $\Phi \circ \Psi = id_{[D^\omega \to D]}$ and $\Psi \circ \Phi = id_D$.

(3) The interpretation function $[\![M]\!]_{\rho,\theta}$ is inductively well-defined for $\lambda\mu$-terms M, mappings $\rho : \mathrm{Vars}_T \to D$, and $\theta : \mathrm{Vars}_S \to D^\omega$ as follows:

$$[\![x]\!]_{\rho,\theta} = \rho(x)$$
$$[\![\lambda x.M]\!]_{\rho,\theta} = \Psi(\overline{\lambda} d :: s \in D^\omega.\Phi([\![M]\!]_{\rho[x \mapsto d],\theta})(s))$$
$$[\![(M)N]\!]_{\rho,\theta} = \Psi(\overline{\lambda} s \in D^\omega.\Phi([\![M]\!]_{\rho,\theta})([\![N]\!]_{\rho,\theta} :: s))$$
$$[\![\mu\alpha.M]\!]_{\rho,\theta} = \Psi(\overline{\lambda} s \in D^\omega.[\![M]\!]_{\rho,\theta[\alpha \mapsto s]})$$
$$[\![(M)\alpha]\!]_{\rho,\theta} = \Phi([\![M]\!]_{\rho,\theta})(\theta(\alpha)).$$

Here, $\rho[x \mapsto d]$ is defined by

$$\rho[x \mapsto d](y) = \begin{cases} d & (x = y) \\ \rho(y) & (x \neq y), \end{cases}$$

and $\theta[\alpha \mapsto s]$ is defined similarly.

The condition (3) requires that each argument of Ψ is contained in $[D^\omega \to D]$.

Theorem 2. (Soundness) Let D be an arbitrary stream model with the interpretation function $[\![\cdot]\!]$. If $M =_{\lambda\mu} N$, then $[\![M]\!]_{\rho,\theta} = [\![N]\!]_{\rho,\theta}$ for any ρ and θ.

4.2 Algebraic Characterization

We give a syntax-free characterization of the extensional stream models.

Definition (Stream combinatory algebras). (1) For a non-empty set D, $\langle D; \cdot, \star \rangle$ is called a *stream applicative structure* if $\cdot : D \times D \to D$ and $\star : D \times D^\omega \to D$ are mappings such that

$$d_1 \star (d_2 :: s_3) = d_1 \cdot d_2 \star s_3$$

holds for any $d_1, d_2 \in D$ and $s_3 \in D^\omega$.

(2) A stream applicative structure $\langle D; \cdot, \star \rangle$ is *extensional* if the following holds for any $d, d' \in D$:

$$\forall s \in D^\omega[d \star s = d' \star s] \text{ implies } d = d'.$$

(3) A stream applicative structure $\langle D; \cdot, \star \rangle$ is called a *stream combinatory algebra* if D contains distinguished elements k_0, k_1, s_0, s_1, $\mathsf{c}_{1,0}$, $\mathsf{c}_{1,1}$, and w_1 such that the following hold for any $d_1, d_2, d_3 \in D$ and $s_2, s_3 \in D^\omega$.

$$\mathsf{k}_0 \cdot d_1 \cdot d_2 = d_1$$
$$\mathsf{k}_1 \cdot d_1 \star s_2 = d_1$$
$$\mathsf{s}_0 \cdot d_1 \cdot d_2 \cdot d_3 = d_1 \cdot d_3 \cdot (d_2 \cdot d_3)$$
$$\mathsf{s}_1 \cdot d_1 \cdot d_2 \star s_3 = d_1 \star s_3 \cdot (d_2 \star s_3)$$
$$\mathsf{c}_{1,0} \cdot d_1 \star s_2 \cdot d_3 = d_1 \cdot d_3 \star s_2$$
$$\mathsf{c}_{1,1} \cdot d_1 \star s_2 \star s_3 = d_1 \star s_3 \star s_2$$
$$\mathsf{w}_1 \cdot d_1 \star s_2 = d_1 \star s_1 \star s_2$$

A stream applicative structure $\langle D; \cdot, \star \rangle$ can be seen as an infinitary algebra with a binary operator $\cdot$ and an ω-ary operator $\bullet$, in which $\star$ is defined as $d \star (d_1, d_2 \cdots) = \bullet(d, d_1, d_2, \cdots)$. Then, roughly speaking, the condition for the stream applicative structure means that $\bullet$ is the extension of the binary operation $\cdot$ to ω-sequences, that is, $\bullet(d_1, d_2, d_3 \cdots) = d_1 \cdot d_2 \cdot d_3 \cdots$.

It is clear that any stream combinatory algebra $\langle D; \cdot, \star \rangle$ is always a combinatory algebra by ignoring the stream part, that is, $\star$, k_1, s_1, $\mathsf{c}_{1,0}$, $\mathsf{c}_{1,1}$, and w_1. Moreover, if D is extensional as a stream applicative structure, then it is extensional as an applicative structure. Indeed, if we suppose $d_1 \cdot d = d_2 \cdot d$ for any $d \in D$, then for any $s \in D^\omega$ we have $d_1 \cdot d \star s = d_2 \cdot d \star s$, which means $d_1 \star (d :: s) = d_2 \star (d :: s)$ for any d and s. Hence, by the extensionality of stream applicative structures,

we have $d_1 = d_2$. Therefore, any extensional stream combinatory algebra is an extensional combinatory algebra, and hence an extensional λ-model.

Definition (Interpretation of CL^{μ}_{ext}). Let $\langle D; \cdot, \star\rangle$ be a stream combinatory algebra. For CL^{μ}_{ext}-terms T, mappings $\rho : \mathrm{Vars}_T \to D$, and $\theta : \mathrm{Vars}_S \to D^\omega$, the mapping $[T]_{\rho,\theta}$ is defined by:

$$\begin{aligned}[C]_{\rho,\theta} &= c\\ [x]_{\rho,\theta} &= \rho(x)\\ [T \cdot U]_{\rho,\theta} &= [T]_{\rho,\theta} \cdot [U]_{\rho,\theta}\\ [T \star \alpha]_{\rho,\theta} &= [T]_{\rho,\theta} \star \theta(\alpha),\end{aligned}$$

where c denotes the element of D corresponding to the constant C, that is, $[\mathsf{K}_0]_{\rho,\theta} = \mathsf{k}_0$, $[\mathsf{S}_0]_{\rho,\theta} = \mathsf{s}_0$, and so on.

Theorem 3. Let $\langle D; \cdot, \star\rangle$ be an extensional stream combinatory algebra. If $T =_{\mathsf{CL}^{\mu}_{ext}} U$, then $[T]_{\rho,\theta} = [U]_{\rho,\theta}$ for any ρ and θ.

Theorem 4. For a non-empty set D, the following are equivalent.

(1) D is an extensional stream model with some Φ and Ψ.

(2) D is an extensional combinatory algebra with some elements k_0, k_1, s_0, s_1, $\mathsf{c}_{1,0}$, $\mathsf{c}_{1,1}$, w_1 in D, and some operations $\cdot$ and $\star$.

Proof. ((1)$\Longrightarrow$(2)) Suppose $\langle D; \Phi, \Psi\rangle$ is an extensional stream model. Define

$$\begin{aligned}d \star s &= \Phi(d)(s)\\ d \cdot d' &= \Psi(\overline{\lambda} s \in D^\omega.\Phi(d)(d' :: s)),\end{aligned}$$

where we should note that $d \cdot d'$ is identical to $[\![xy]\!]_{\rho[x\mapsto d, y\mapsto d']}$ and hence it is always defined, and define

$$\begin{aligned}\mathsf{k}_0 &= [\![\lambda xy.x]\!]\\ \mathsf{k}_1 &= [\![\lambda x.\mu\alpha.x]\!]\\ \mathsf{s}_0 &= [\![\lambda xyz.xz(yz)]\!]\\ \mathsf{s}_1 &= [\![\lambda xy.\mu\alpha.x\alpha(y\alpha)]\!]\\ \mathsf{c}_{1,0} &= [\![\lambda x.\mu\alpha.\lambda y.xy\alpha]\!]\\ \mathsf{c}_{1,1} &= [\![\lambda x.\mu\alpha\beta.x\beta\alpha]\!]\\ \mathsf{w}_1 &= [\![\lambda x.\mu\alpha.x\alpha\alpha]\!],\end{aligned}$$

where all the $\lambda\mu$-terms in the right-hand sides contain no free variable, so they are independent of ρ and θ. Then $\langle D; \cdot, \star\rangle$ is an extensional stream combinatory algebra. Indeed, it is a stream applicative structure, since

$$\begin{aligned}d_1 \cdot d_2 \star s_3 &= \Phi(\Psi(\overline{\lambda} s.\Phi(d_1)(d_2 :: s)))(s_3)\\ &= \Phi(d_1)(d_2 :: s_3)\\ &= d_1 \star (d_2 :: s_3).\end{aligned}$$

((2)$\Longrightarrow$(1)) Suppose $\langle D; \cdot, \star\rangle$ is an extensional stream combinatory algebra. Define $[D^\omega \to D] := \{f_d \mid d \in D\}$, where f_d is $\overline{\lambda} s \in D^\omega.d \star s$, and

$$\Phi(d) = f_d \qquad \Psi(f_d) = d.$$

Note that Φ and Ψ is well-defined since D is extensional. If $f_d = f_{d'}$, we have $f_d(s) = f_{d'}(s)$ for any $s \in D^\omega$, that is, $d \star s = d' \star s$. Hence we have $d = d'$ by the extensionality.

Then the interpretation $[\![\cdot]\!]$ can be defined as $[\![M]\!]_{\rho,\theta} = [M^*]_{\rho,\theta}$. In order to show that, we use the following lemmas:

$$\begin{aligned}[\lambda^* x.T]_{\rho,\theta} \cdot d &= [T]_{\rho[x\mapsto d],\theta}\\ [\mu^* \alpha.T]_{\rho,\theta} \star s &= [T]_{\rho,\theta[\alpha\mapsto s]}.\end{aligned}$$

For example, in the case of $M = \lambda x.N$, we have

$$\begin{aligned}[M^*]_{\rho,\theta} \star (d :: s) &= [M^*]_{\rho,\theta} \cdot d \star s\\ &= [N^*]_{\rho[x\mapsto d],\theta} \star s\\ &= [\![N]\!]_{\rho[x\mapsto d],\theta} \star s\\ &= \Phi([\![N]\!]_{\rho[x\mapsto d],\theta})(s)\end{aligned}$$

for any d and s by the lemma and the induction hypothesis. Since

$$\begin{aligned}\overline{\lambda} d :: s.[M^*]_{\rho,\theta} \star (d :: s) &= \overline{\lambda} s.[M^*]_{\rho,\theta} \star s\\ &= f_{[M^*]_{\rho,\theta}},\end{aligned}$$

we have $\overline{\lambda} d :: s.\Phi([\![N]\!]_{\rho[x\mapsto d],\theta})(s) = f_{[M^*]_{\rho,\theta}}$, which is an element of $[D^\omega \to D]$. Therefore, $[\![M]\!]_{\rho,\theta}$ is defined and identical to $\Psi(f_{[M^*]_{\rho,\theta}}) = [M^*]_{\rho,\theta}$. The other cases are similarly proved.

Hence, $\langle D; \Phi, \Psi\rangle$ is an extensional stream model.

5 Conclusion

Summary. We have proposed the new combinatory calculus CL^{μ}_{ext}, which exactly corresponds to the untyped $\lambda\mu$-calculus. Terms of CL^{μ}_{ext} can be seen as combinators for both data and streams. We have also proposed the extensional stream models of the untyped $\lambda\mu$-calculus, and have shown that the extensional stream models are algebraically characterized independently of the syntax of the $\lambda\mu$-calculus.

In this paper, we formalize CL^{μ}_{ext} with the seven constants, but it just means that the seven constants are sufficient. (As noted below, the choice of the indexed constants is suitable for the generalization to the stream hierarchy.) Remind that the combinators C and W in CL are definable from K

and S. It is an interesting question whether some of the constants of CL^{μ}_{ext} are definable from the others.

(Extensional) stream models. We have not shown any concrete extensional stream model in this paper, but we can obtain (non-trivial) models as solutions of the domain equation $D \simeq D^{\omega} \to D$ in an appropriate category such as $\mathrm{CPO}^{\mathrm{E}}$ of embeddings on CPOs. Let B be a non-trivial domain, C be B^{ω}, and D be a solution of $D \simeq D \to C$. Then we have $C^{\omega} \simeq B^{\omega\times\omega} \simeq B^{\omega} = C$, and therefore $D \simeq D \to C \simeq D \to C^{\omega} \simeq (D \to C)^{\omega} \simeq D^{\omega}$. Hence, we have $D \simeq D^{\omega} \to C \simeq D^{\omega+1} \to C \simeq D^{\omega} \to (D \to C) \simeq D^{\omega} \to D$.

In particular, we can construct extensional stream models in a similar way to Scott's D_{∞} [7] [8]. For any CPO D, we can find a stream model D^{μ}_{∞} into which D can be embedded. However, we have not understood D^{μ}_{∞} sufficiently: Does D^{μ}_{∞} enjoy the approximation theorem? Which syntactic equality corresponds to the equality in D^{μ}_{∞}? How does it related to the Böhm-tree semantics in [6]?

We can also generalize the class of stream models to non-extensional ones. The definition of the *stream models* is the same as the extensional stream models except for the condition $\Psi \circ \Phi = id$. In such stream models, however, the axiom (β_T) is preserved only for the restricted form:

$$(\lambda x.M)N_1 \cdots N_n\alpha =_{\beta_T} M[x := N_1]\cdots N_n\alpha.$$

Hence, if we restrict the $\lambda\mu$-terms to

$$M ::= x \mid \lambda x.M \mid \mu\alpha.M \mid (M)M_1 \cdots M_n\alpha,$$

the calculus is sound with respect to the stream models.

Generalizing to stream hierarchy. Saurin [5] proposes a further generalization of $\lambda\mu$-calculus, called the *stream hierarchy*, in which we can treat not only streams of data, but also streams of streams and streams of streams of streams, and so on. Moreover, Saurin shows that the stream hierarchy corresponds the call-by-name variant of the CPS hierarchy.

The stream hierarchy is a collection of calculi $(\Lambda^n)_{n\in\omega}$ indexed with natural numbers. In particular, Λ^0 is the ordinary λ-calculus, and Λ^1 is the $\lambda\mu$-calculus. The combinatory calculus CL^{μ}_{ext} can also be extended to the stream hierarchy. We can construct the collection of combinatory calculi $(\mathsf{CL}^n)_{n\in\omega}$, in which CL^0 is the ordinary (untyped) combinatory logic, and CL^1 is CL^{μ}_{ext}. In fact, the calculus CL^n is defined with the constants $\langle \mathsf{K}_i, \mathsf{S}_i, \mathsf{C}_{j,i}, \mathsf{W}_j \mid 0 \le i \le n, 1 \le j \le n\rangle$. Note that this representation includes the cases of $\mathsf{CL}(= \mathsf{CL}^0)$ and $\mathsf{CL}^{\mu}_{ext}(= \mathsf{CL}^1)$: the set of constants is $\{\mathsf{K}_0, \mathsf{S}_0\}$ when $n = 0$, and $\{\mathsf{K}_0, \mathsf{K}_1, \mathsf{S}_0, \mathsf{S}_1, \mathsf{C}_{1,0}, \mathsf{C}_{1,1}, \mathsf{W}_1\}$ when $n = 1$. Moreover, the notion of the extensional stream models is also extended to Λ^n by changing the equality $D \simeq [D^{\omega} \to D]$ to $D \simeq [D^{\omega^n} \to D]$, and the results in this paper are generalized to the stream hierarchy.

Models of the untyped $\lambda\mu$-calculus. In [9], Streicher and Reus proposes the continuation models for the untyped $\lambda\mu$-calculus, which is based on the idea that the $\lambda\mu$-calculus is a calculus of continuations. If we see each stream $d :: s$ as a pair $\langle d, s\rangle$ of an argument d of function and a continuation s, the interpretation function for the stream models looks exactly the same as that for the continuation models. The main difference between them is that the continuation models are only for (a variant of) the Parigot's original $\lambda\mu$-calculus, in which the named terms are distinguished from the ordinary terms. In the continuation models, ordinary terms are interpreted as functions from continuations to responses (i.e. results of computations), whereas named terms are interpreted as responses. Hence, for example, a term of the form $(M)\alpha N$ cannot be interpreted in the continuation models. On the other hand, in the extensional stream models, both terms and named terms are uniformly interpreted in D.

Akama [1] shows that the untyped $\lambda\mu$-calculus can be interpreted in partial combinatory algebras. It is based on the idea that μ-abstractions are functions on streams. However, it restricts terms to affine ones, that is, each bound variable must not occur more than once.

Fujita [2] considers a reduction system for the $\lambda\mu$-calculus with (β_T), (η_T), (μ), and (fst) rules, and gives a translation from the $\lambda\mu$-calculus to the λ-calculus which preserves the equality, and hence it is shown that any extensional λ-model is a model of the $\lambda\mu$-calculus. In the translation, each μ-abstraction is interpreted as a potentially infinite λ-abstraction by means of a fixed point operator. However, it considers neither (β_S) nor (η_S), and it seems hard to obtain a similar result for them. Every extensional stream model is an extensional λ-model, but it should be further studied how the extensional stream models relate to the λ-models:

Is there any λ-model which cannot be a stream model?

Acknowledgments I am grateful to Dana Scott for helpful comments, and to Shin-ya Katsumata and Daisuke Kimura for fruitful discussions.

References

[1] Akama, Y. Limiting partial combinatory algebras towards infinitary. In *Proceedings of Computer Science Logic (CSL 2001)*, volume 2142 of *LNCS*, pages 399–414, 2001.

[2] Fujita, K. An interpretation of $\lambda\mu$-calculus in λ-calculus. *Information Processing Letters*, 84:261–264, 2002.

[3] Parigot, M. $\lambda\mu$-calculus: an algorithmic interpretation of classical natural deduction. In *Proceedings of the International Conference on Logic Programming and Automated Reasoning (LPAR '92)*, volume 624 of *LNCS*, pages 190–201, 1992.

[4] Saurin, A. Separation with streams in the $\lambda\mu$-calculus. In *20th Annual IEEE Symposium on Logic in Computer Science (LICS' 05)*, pages 356–365, 2005.

[5] Saurin, A. A hierarchy for delimited control in call-by-name. In *13th International Conference on Foundations of Software Science and Computation Structures (FoSSaCS 2010)*, volume 6014 of *LNCS*, pages 374–388, 2010.

[6] Saurin, A. Standardization and böhm trees for $\lambda\mu$-calculus. In *Tenth International Symposium on Functional and Logic Programming (FLOPS 2010)*, volume 6009 of *LNCS*, pages 134–149, 2010.

[7] Scott, D.S. Continuous lattices. In *Toposes, Algebraic Geometry, and Logic*, volume 274 of *Lecture Notes in Mathematics*, pages 97–136. 1972.

[8] Smyth, M.B. and Plotkin, G.D. The category-theoretic solution of recursive domain equations. *SIAM Journal on Computation*, 11(4):761–783, 1982.

[9] Streicher, T. and Reus, B. Classical logic, continuation semantics and abstract machines. *Journal of Functional Programming*, 8(6):543–572, November 1998.